Bio101 Development Document

1. Introduction

1.1 Overview

Bio101 is a software to encode any file to DNA sequences and decode the DNA sequences to the original file so that everyone can store any information in real DNA sequences.

1.2 Definitions and abbreviations

ISAAC: This is an encryption algorithm. You can view this page (http://burtleburtle.net/bob/rand/isaacafa.html) to know more about ISAAC.

SBOL: The full name is The Synthetic Biology Open Language. SBOL provides a community standard for communicating designs about synthetic biology.

IDE: The full name is Integrated Development Environment. Generally, IDE includes code editor, compiler, debugger, and graphical user interface tools. Integrating code writing function with analysis function, compile function, debug function completes the development of software service sets.

1.3 Environment and platform

CPU：Intel core i5 3230M 2.6GHz

Motherboard：Intel HM76

Cache：DDR3L 1600 MHz 4GB

System：Windows 10

Hard disk：WESTERN DIGITAL (WD7500BPKX) 750GB

Graphics cards: NVIDIA GeForce GT 720M

Network card：RealtekRTL8111

1.4 Tools

We mainly use Python to accomplish our software, and the editor we used is Sublime text3.

2. Detail process

2.1 Archival mode

To store information into DNA sequences, we do several steps briefly as follow*(Fig.1)*. Firstly, in order to reduce the length of the information and preserve attribute of the file, we compress the file by the bz2 algorithm. Secondly, we encrypt the file depending on ISAAC, the secret code. Thirdly, we convert the encrypted file into quaternary and then convert it to the DNA sequence. Fourthly, we make the DNA sequence four times-fold redundancy. Lastly, we add indexes to each of the sub DNA sequences for decoding the DNA sequences correctly. C-language module Isaac, bit2nt and nt2bit is used to encode and decode information.

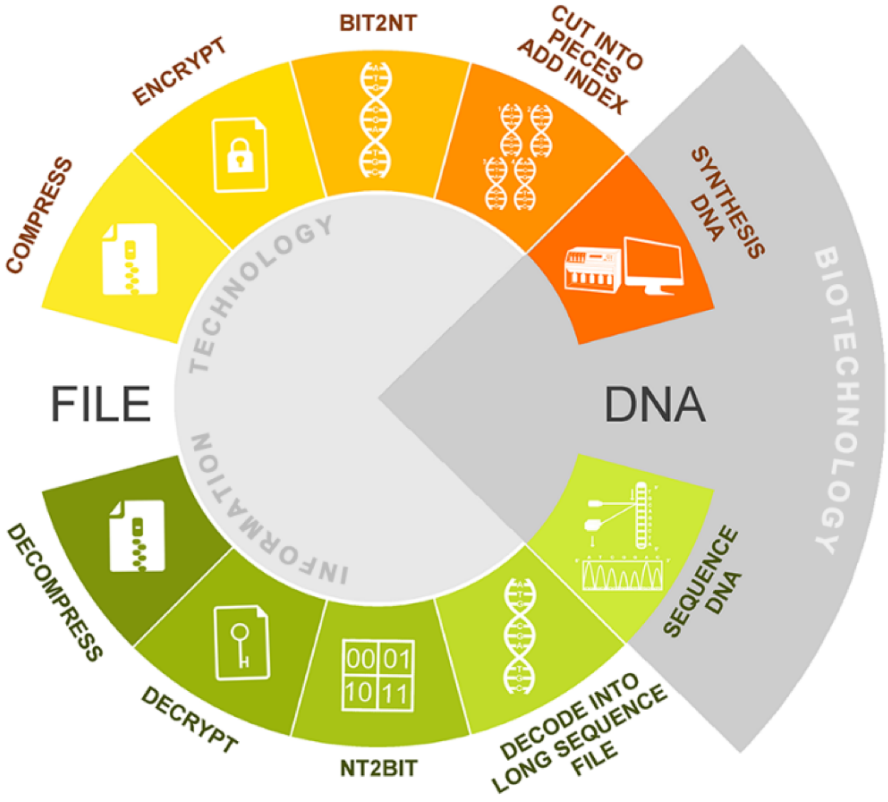

Fig.1. The whole process of encoding and decoding of archival mode.

2.2 Editable mode

In order to make our storage system to be editable, we use a more flexible method to encode information. We cut the integrate file into 32 byte per part at first and encrypt and random it by Isaac, then we convert these parts to DNA fragments. After that, we add index to each DNA fragment and complete encode. In order to design sgRNA, a PAM site (5'-NGG-3') is added after the index code. User can decode a single DNA fragment to edit and then encode it again to get a new DNA fragment. C-language module isbit2nt and isnt2bit is used to encode and decode information in this model, and Isaac is integrate into isbit2nt and isnt2bit.

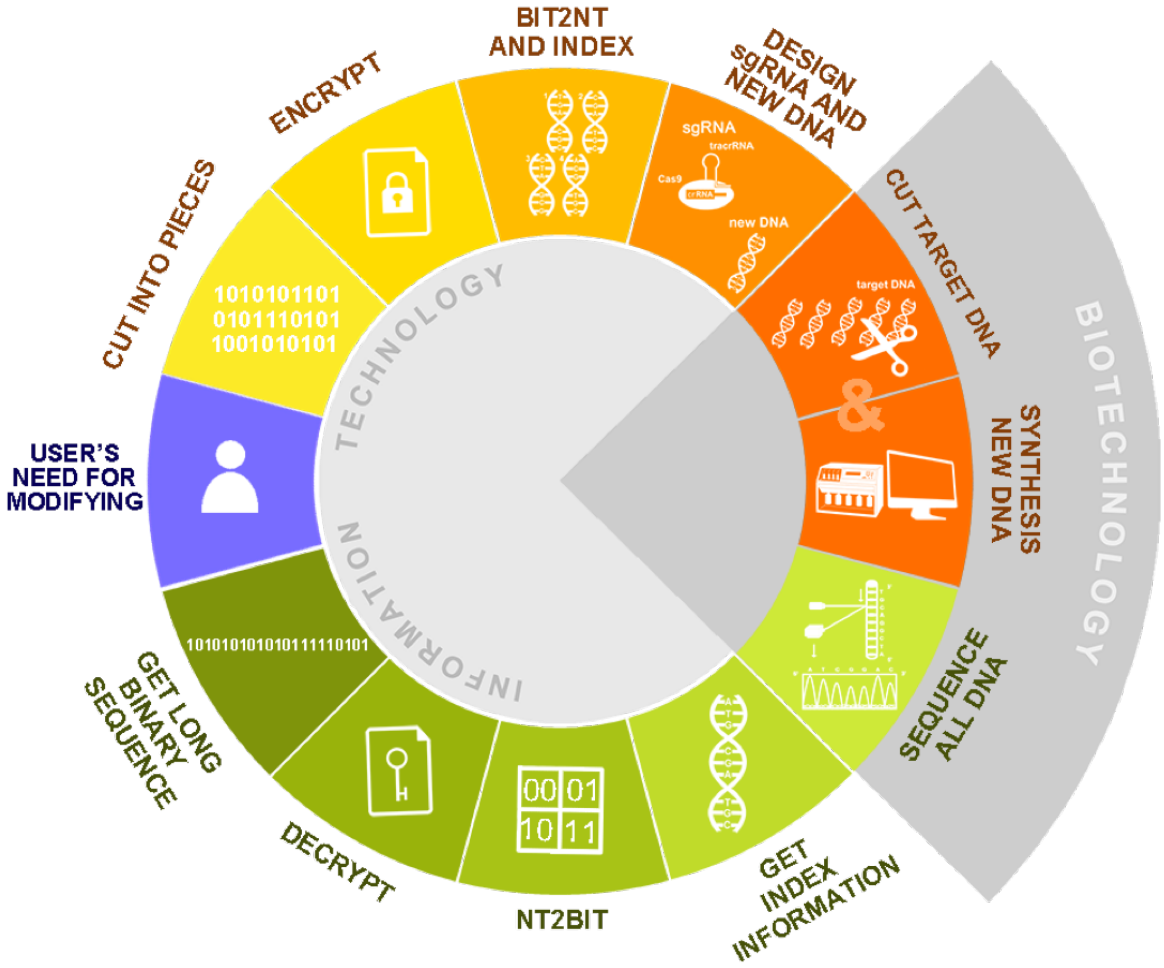

Fig.2. The whole process of encoding and decoding of editable mode.

2.3 sgRNA design

After information is edited, a guide gene is designed to code sgRNA and transcribe sgRNA to instruct Cas9 to degrade the DNA fragment which need to be delete. A xml template is used in this part. The guid gene is consist of a pBAD promoter, a guide region, a Cas9 handle and a S. pyogenes terminator. In which, the guide region is a template from the target DNA.

3. Functions

3.1 def convert_file_to_DNA (path)

This function is to convert a file to original DNA sequence. Firstly compressing the file to zip package, secondly encrypting the zip package by ISAAC algorithm and converting it to quaternary, thirdly converting the quaternary to original DNA sequence by 0 to A, 1 to C, 2 to G and 3 to T. It aims to form the original DNA sequence.

3.2 def convert_DNA_to_file (path)

This function is to convert a DNA sequence to file, converting DNA to bit and decrypting by ISAAC algorithm, then generating a zip package, at last, decompressing the zip package to get the original file.

3.3 def to DNA (string)

All information is added to original sequence and sub sequence is quaternary. We convert the quaternary to DNA by 0 to A, 1 to C, 2 to G and 3 to T firstly. And then we make the DNA random following the rule in table.1. (The first character is unchanged. For the following sequence, characters are taken from the row defined by the previous changed character conversion.)

previous/next	A	C	G	T
A	C	G	T	A
C	G	T	A	C
G	T	A	C	G
T	A	C	G	T

Table.1 Random process.

3.4 def to_NUM (dna)

This function is to record the DNA sequence using the following table and then convert the ordered DNA sequence to quaternary by 0 to A, 1 to C, 2 to G and 3 to T. (The first character is unchanged. For the following sequence, characters are taken from the row defined by the previous character conversion.)

previous/next	A	C	G	T
A	T	A	C	G
C	G	T	A	C
G	C	G	T	A
T	A	C	G	T

Table.2. Record DNA Sequence.

3.5 def encoding_file (request, seq, fname, ftype)

It is to write the sub sequences into a file according to the file type given by users which includes txt, fasta, and SBOL-Xml.

3.6 def reverse_s (string)

This function is to reverse a string and return the reversed string.

3.7 check (dna_list, length)

Use fuzzy algorithm to check the DNA sequences in the dna_list and return a correct string.

3.8 def file_seq(file)

This function is to get all sequences from a file. It accepts file type including txt, fasta and xml format. The txt file contains pure DNA sequences, and Xml is in the standard of SBOL. They all return a list of all DNA sequences.

3.9 def __init__ (self, path)

This function is the working function. It calls other functions to convert file to final DNA sequences. It runs with the following steps.

1. Add length information to the end of the original DNA sequence and make sure the sequence length as the multiple of 50.

2. Split the DNA sequence to units with 50bp per unit, join four units to a sub sequence and make it four times-fold redundancy. And then, reverse odd units.

3. Add index and check of index information to sub sequences.

4. Add A or T to the head of sub sequences and C or G to the tail of sub sequences to label the order of sub sequences.

3.10 def make_S5 (self)

Count the length of the original DNA sequence and turn it to quaternary(S2), add “0” (S3) before S2 to ensure the total length of original DNA sequence, S3 and S2 is the multiples of 50. (The order is original DNA sequence, S3, S2. S2 and S3 are converted to DNA by ‘toDNA()’ function). Return the treated DNA sequence as S5.

3.11 def make_F (self)

This is function is to convert sub sequences to file by the following steps.

1. Get the sequences from a file.

2. Get the index of sub sequences and P, check the index by parity-check. Then, order the sub sequences by analyzing the start with A or T and end with C or G.

3. Check the sub sequences which have the same index by fuzzy algorithm and get the correct sub sequence of each index.

4. Split the sub sequences into 50bp per units and check the corresponding units in different sub sequences by fuzzy algorithm and get the correct unit.

5. Connect the units and get the length information of original DNA sequence.

6. Get the original sequence and write it into a text file.

7. Convert the original DNA sequence to bit by nt2bit, then decrypt it and convert it to zip package by ISAAC.

Split the DNA sequence to units with 50bp per unit, add four units to a sub sequence and make it four times-fold redundancy. Then, reverse odd units. Return a list(F) which contains the primary sub sequences.

3.12 def make_subSequence(self)

Add index and check information of index(P) to every sub sequence, then add A or T to the head of sub sequences and C or G to the tail. Return a list which contains the final sub sequences.

3.13 def __init__ (self, path, fname)

This is function is to convert sub sequences to file by the following steps.

1. Get the sequences from a file.

2. Get the index of sub sequences and P, check the index by parity-check. Then, order the sub sequences by analyzing the start with A or T and end with C or G.

3. Check the sub sequences which have the same index by fuzzy algorithm and get the correct sub sequence of each index.

4. Split the sub sequences into 50bp per units and check the corresponding units in different sub sequences by fuzzy algorithm and get the correct unit.

5. Connect the units and get the length information of original DNA sequence.

6. Get the original sequence and write it into a text file.

7. Convert the original DNA sequence to bit by nt2bit, then decrypt it and convert it to zip package by ISAAC.

8. Decompress the zip package and get the original file.

3.14 def get_Fi (sequence)

Get the index of sub sequences and P, check the index by parity-check. Then, order the sub sequences by analyzing the start with A or T and the end with C or G. Check the subsequences which have the same index by fuzzy algorithm and get the correct sub sequence of each index. Return it as Fi.

3.15 def get_seq(F)

Split the sub sequences into 50bp per units, check the corresponding units in different sub sequences by fuzzy algorithm and get the correct unit. Connect the units and get the length information of original DNA sequence. Get the original sequence and write it into a text file.

3.16 def edit()

User could submit a DNA fragment and process will decode it by Editable mode and return the decoded information.

3.17 def edited()

User submit the edited information which decoded form the DNA fragment just submitted. Software will encode it again and return the original DNA fragment and new fragment, and then design a guide DNA according to the original fragment. And the guide DNA will display as a SBOL file to provide user to download it.